Q1

(a) (5 points) (written)

i. (2 points) Provide 2 examples of sentences containing a named entity with an ambiguous type (e.g. the entity could either be a person or an organization, or it could either be an organization or not an entity).

ii. (1 point) Why might it be important to use features apart from the word itself to predict named entity labels?

iii. (2 points) Describe at least two features (apart from the word) that would help in predicting whether a word is part of a named entity or not.

(b) (5 points) (written)

i. (2 points) What are the dimensions of $\mathbf{e}^{(t)}$, W and U if we use a window of size w?

ii. (3 points) What is the computational complexity of predicting labels for a sentence of length T?

a)

i. "Robert Walters is financially strong", could be a person or the organization. "Apple is good", could be an apple, the tech giant Apple or its product.

ii. Often, named entities can be rare words. Using casing helps the system to generalize.

iii. Casing of words and parts of speech.

b)

i. $e^{(t)} \in R^{(2w+1)D}, W \in R^{(2w+1)D\times H}, U \in R^{H\times C}$

ii. $O(\, T(\,(2w+1)V \cdot D + (2w+1)D \cdot H + H \cdot C)\,)$, mainly from matrix multiplications.

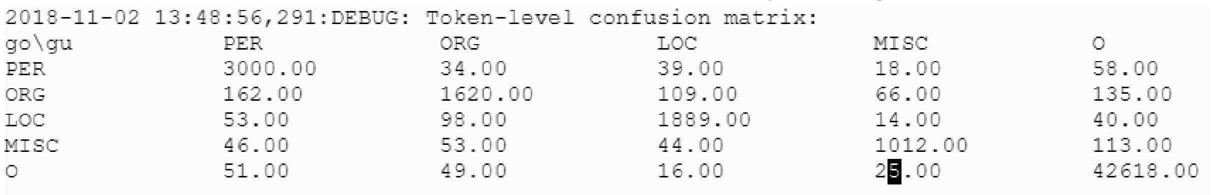
(d) (5 points) (written) Analyze the predictions of your model using the files generated above.

i. (1 point) Report your best development entity-level F_1 score and the corresponding token-level confusion matrix. Briefly describe what the confusion matrix tells you about the errors your model is making.

ii. (4 points) Describe at least 2 modeling limitations of the window-based model and support these conclusions using examples from your model's output (i.e. identify errors that your model made due to its limitations). You can also support your conclusions using predictions made by your model on examples manually entered through the shell.

d)

i. Best dev-F_1 = 0.84. The diagonals of the confusion matrix tells us how many samples are classified correctly for each class. The off-diagonals tells us how many samples are classified wrongly, with the row representing the wrong prediction and the column the ground truth for those samples. Seems like the model predicts well for PER classes but makes the most mistakes when predicting ORG classes.



```
2018-11-02 13:48:56,291:DEBUG: Token-level confusion matrix:
go\gu           PER             ORG             LOC             MISC            O
PER             3000.00         34.00           39.00           18.00           58.00
ORG             162.00          1620.00         109.00          66.00           135.00
LOC             53.00           98.00           1889.00         14.00           40.00
MISC            46.00           53.00           44.00           1012.00         113.00
O               51.00           49.00           16.00           25.00           42618.00
```

ii. Window-based NER model cannot use information from neighbouring window predictions to enhance its classification results. Eg. "Watching the Hong Kong Open on Friday". The window [the Hong Kong] could classify as LOC or ORG, but [Hong Kong Open] most likely results in ORG and so does [Kong Open on]. It also cannot use information from other parts of the sentence. Eg. "'I am rich', said the beautiful Paris who… ". Here, Paris could be a LOC or a PER.

2. Recurrent neural nets for NER (40 points)

We will now tackle the task of NER by using a recurrent neural network (RNN). See slide 47 ('RNNs can be used for tagging') of Lecture 8 for an illustration of this kind of tagging task. Recall that each RNN cell combines the previous hidden state with the current input using a sigmoid. We then use the hidden state to predict the output at each timestep:

$$\mathbf{e}^{(t)} = \mathbf{x}^{(t)}E$$
$$\mathbf{h}^{(t)} = \sigma(\mathbf{h}^{(t-1)}W_h + \mathbf{e}^{(t)}W_e + \mathbf{b}_1)$$
$$\hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{h}^{(t)}U + \mathbf{b}_2),$$

where $E \in \mathbb{R}^{V\times D}$ are word embeddings, $W_h \in \mathbb{R}^{H\times H}$, $W_e \in \mathbb{R}^{D\times H}$ and $\mathbf{b}_1 \in \mathbb{R}^{H}$ are parameters for the RNN cell, and $U \in \mathbb{R}^{H\times C}$ and $\mathbf{b}_2 \in \mathbb{R}^{C}$ are parameters for the softmax. As before, V is the size of the vocabulary, D is the size of the word embedding, H is the size of the hidden layer and C are the number of classes being predicted (here 5).

In order to train the model, we use a cross-entropy loss for the every predicted token:

$$J = \sum_{t=1}^{T} CE(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)})$$
$$CE(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)}) = -\sum_{i} y_i^{(t)} \log(\hat{y}_i^{(t)}).$$

(a) (4 points) (written)

i. (1 point) How many more parameters does the RNN model in comparison to the window-based model?

ii. (3 points) What is the computational complexity of predicting labels for a sentence of length T (for the RNN model)?

(b) (2 points) (written) Recall that the actual score we want to optimize is entity-level F_1.

i. (1 point) Name at least one scenario in which decreasing the cross-entropy cost would lead to an *decrease* in entity-level F_1 scores.

ii. (1 point) Why it is difficult to directly optimize for F_1?

a)

i) Window-based $= V\times D\ +\ (2w+1)\times D\times H\ +\ H\ +\ H\times C\ +\ C.$
RNN-based $= V\times D + H\times H + D\times H + H + H\times C + C.$
RNN has $\frac{H}{2wD}$ more parameters.

ii) $O(T(V\times D + H\times H + D\times H + H\times C))$

b)

i) Eg. "The Spacey Scandal", with gold labels "O MISC MISC". If the model predicts "O O O" previously but now predicts "O MISC O", the cross entropy loss would reduce since one more token has a correct prediction. However, precision reduces since you have one extra FP ("Spacey Scandal" is one entity, if just "Spacey" is labelled as an entity, it is a FP, since no entity with just "Spacey" is found in the golden reference; the model predicts an extra entity which is considered a FP). Recall is the same since there are no new FN (as long as "Spacey Scandal" is not labelled as "MISC MISC" it will trigger an FN). This makes F_1 drop.

ii) F_1 is not differentiable. It also needs all predictions from the whole corpus to compute, so it is hard to parallelize and make batches to optimize.

(d) (8 points) (code/written) Implementing an RNN requires us to unroll the computation over the whole sentence. Unfortunately, each sentence can be of arbitrary length and this would cause the RNN to be unrolled a different number of times for different sentences, making it impossible to batch process the data.

The most common way to address this problem is *pad* our input with zeros. Suppose the largest sentence in our input is M tokens long, then, for an input of length T we will need to:

1. Add "0-vectors" to $\mathbf{x}$ and $\mathbf{y}$ to make them M tokens long. These "0-vectors" are still one-hot vectors, representing a new `NULL` token.
2. Create a *masking vector*, $(m^{(t)})_{t=1}^{M}$ which is 1 for all $t \leq T$ and 0 for all $t > T$. This masking vector will allow us to ignore the predictions that the network makes on the padded input.[3]
3. Of course, by extending the input and output by $M - T$ tokens, we might change our loss and hence gradient updates. In order to tackle this problem, we modify our loss using the masking vector:

$$J = \sum_{t=1}^{M} m^{(t)} \, \mathrm{CE}(y^{(t)}, \hat{y}^{(t)}).$$

i. (3 points) (written) How would the loss and gradient updates change if we did not use masking? How does masking solve this problem?

d.

i. Loss would be greater and the gradients would tune the weights of the NULL tokens, which is undesired since the padded vectors are dummy examples. The mask ensures no gradient flows back to the NULL token weights and also gives a correct value for the loss since dummy examples are ignored during loss computation.

(g) (6 points) (written)

i. (3 points) Describe at least 2 modeling limitations of this RNN model and support these conclusions using examples from your model's output.

ii. (3 points) For each limitation, suggest some way you could extend the model to overcome the limitation.

g.

i. It cannot use information in the future to help in predicting the present. Eg. "New York University", the RNN would most likely classify 'New' as LOC. The model does not enforce adjacent tokens to have the same class.

ii. Use a bidirectional RNN. Use pair-wise agreement eg. CRF loss.

Q3

(e) (5 points) (written) Analyze the graphs obtained above and describe the learning dynamics you see. Make sure you address the following questions:

i. Does either model experience vanishing or exploding gradients? If so, does gradient clipping help?

ii. Which model does better? Can you explain why?

Important: There is some variance in the graphs you get based on the order of initializing variables. It might be the case that you do not see any effect from clipping gradients – that is ok, report your graphs and what you observe. For reference, the order in which we initialize variables for the GRU is W_r,U_r,b_r,W_z,U_z,b_z,W_o,U_o,b_o and for the RNN cell it is W_h,W_x,b.

e.

i. RNN experience vanishing gradients. So gradient clipping cannot help it (clipping helps exploding gradients). GRU has exploding gradients (see solutions' plot), so clipping can help it.

ii. GRU is better, since the RNN theoretically cannot model latching behaviour. However the loss of RNN is slightly lower (both GRU and RNN have ~12.5 loss)